(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 685 824 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
08.01.1997 Bulletin 1997/02

(43) Date of publication A2:
06.12.1995 Bulletin 1995/49

(21) Application number: **95303532.6**

(22) Date of filing: **24.05.1995**

(51) Int. Cl.[6]: **G06T 9/00**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **31.05.1994 US 251498**

(71) Applicant: **International Business Machines Corporation**
Armonk, N.Y. 10504 (US)

(72) Inventor: **deCarmo, Linden A.**
Plantation, Florida 33324 (US)

(74) Representative: **Moss, Robert Douglas**
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)

(54) Data decompression and transfer system

(57) A system and method for transfer of compressed data that dynamically chooses between hardware and software compression/decompression (CODEC) so as to maximize the usage of any digital signal device hardware capabilities; and that provides a common, uncompressed data interchange format for applications regardless of any compression technique that may have been used to create the data file.

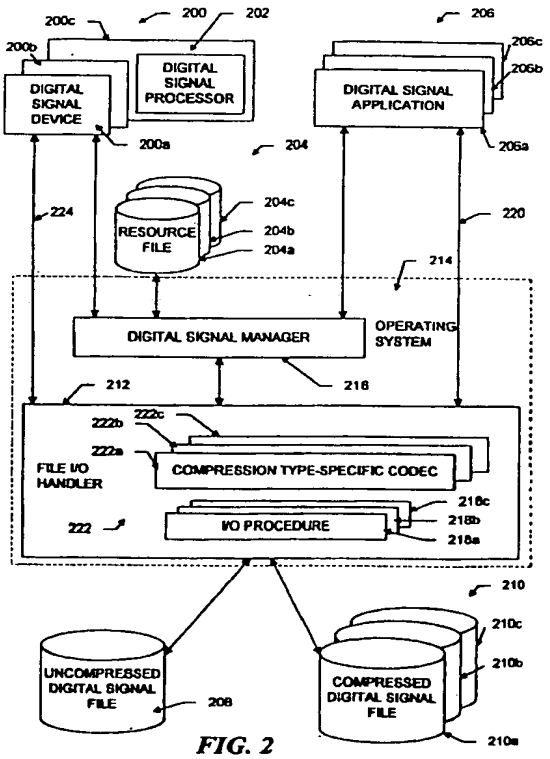

FIG. 2

EP 0 685 824 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 95 30 3532

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	SPEECH TECHNOLOGY, AUG.-SEPT. 1985, USA, vol. 3, no. 1, ISSN 0744-1355, pages 77-79, XP000577297 MORRIS L R: "Real-time spectrograms via hardware/software PC enhancement"	1,2,5-9, 11,12	G06T9/00
Y	* page 79, left-hand column, paragraph 11 - right-hand column, paragraph 1 *	3,4,10, 13	
Y	EP-A-0 512 174 (SEMAPHORE INC) 11 November 1992 * page 3, line 49 - line 55 * * page 5, line 1 - line 5 * * page 6, line 35 - line 41; figure 5 *	3,4,10, 13	
A	ELECTRONIC DESIGN, 9 JULY 1992, USA, vol. 40, no. 14, ISSN 0013-4872, page 40, 42, 44, 46 XP000307168 LEONARD M: "Scanning the options for image compression"		
A	EP-A-0 503 956 (CUBE MICROSYSTEMS C) 16 September 1992		TECHNICAL FIELDS SEARCHED (Int.Cl.6) G06T

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	4 November 1996	Pierfederici, A

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

BNSDOCID: <EP... 0685824A3_I_>

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 685 824 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **95303532.6**

(51) Int. Cl.[6]: **G06T 9/00**

(22) Date of filing: **24.05.95**

(30) Priority: **31.05.94 US 251498**

(43) Date of publication of application:
06.12.95 Bulletin 95/49

(84) Designated Contracting States:
DE FR GB

(71) Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504 (US)

(72) Inventor: **deCarmo, Linden A.**
10467 N.W. Third Street
Plantation,
Florida 33324 (US)

(74) Representative: **Moss, Robert Douglas**
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester
Hampshire SO21 2JN (GB)

(54) Data decompression and transfer system.

(57) A system and method for transfer of compressed data that dynamically chooses between hardware and software compression/decompression (CODEC) so as to maximize the usage of any digital signal device hardware capabilities; and that provides a common, uncompressed data interchange format for applications regardless of any compression technique that may have been used to create the data file.

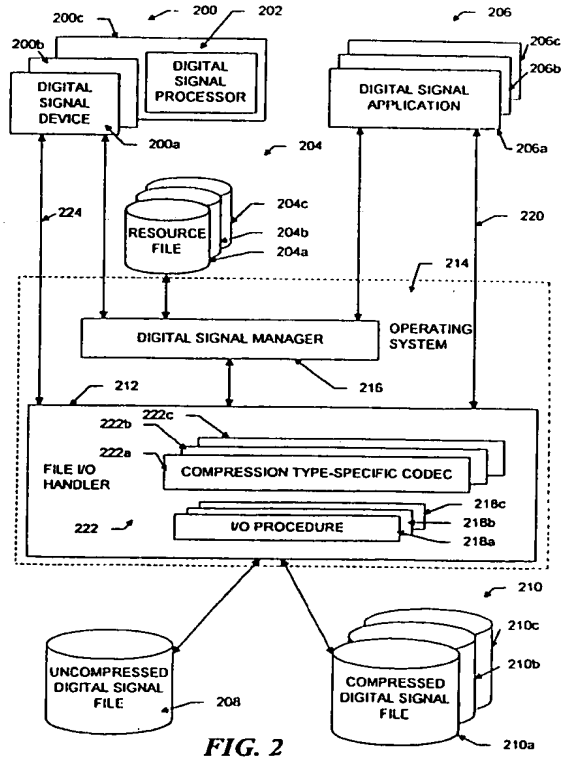

FIG. 2

EP 0 685 824 A2

FIELD OF THE INVENTION

This invention relates generally to decompression and transfer of data, specifically in systems that store and process digitized audio, video, and other signals using one or more compression techniques.

BACKGROUND OF THE INVENTION

Digital signal devices ("devices") are computer peripheral devices that interface between computers and components that process analog signals, for example microphones, speakers, video cameras, or tape recorders. These devices may process a variety of data types including video, graphical, and audio signals. For example, "audio devices" are devices that process audio data. A commonly known type of audio device is a "sound board" connectable to a bus in a computer system. There are a number of commercially available sound boards.

A digital signal device periodically samples an analog input signal and encodes the sampled signal amplitude to generate a stream of digital words, which form "digital signals" or "digitized data". The name associated with the data may reflect its content, for example "audio data". The data can be stored in computer files, for example as a digital representation of music or animation. Digital signals can also be incorporated into files that contain other kinds of data, for example audio data may be incorporated into electronic mail, word processing, or spreadsheet files.

Digital signal applications ("applications") are programs that allow a user to manipulate stored digital signals, for example, to compose music or to cut a segment of audio data from one file and paste it into another file. A number of such digital signal applications are commercially available. Applications and digital signal devices interact with files of stored digital data by means of a file input/output (I/O) handler. The file I/O handler can be part of the computer operating system, and manages the administrative operations of opening and closing files and actually storing and retrieving data in a storage device.

An application can interact with a digital signal device by instructing the device to convert an analog input signal into a digital signal and to store the digital signal in a file ("recording"), or by instructing the device to retrieve a digital signal from a file and to convert the digital signal into an analog output signal ("playback"). Recording and playback are commonly referred to as "real-time" operations because the conversions required should take no longer than the actual time duration of the analog signal. During recording, playback, and other real-time operations, the application does not process the digital data, instead the digital signal device interacts directly with the file.

The increased use of digital audio, video, and other signal processing applications and devices has increased the amount of digital signal data that must be stored and processed by computers. Although the sampling rate and the number of bits per sample determine the actual amount of space required to store a given digital signal, in general, the space required is relatively large. Digital signals may include each bit of each sample, commonly referred to as "pulse code modulated (PCM)" or "uncompressed" signals. Alternately, the signals may be compressed to reduce the required storage space, commonly referred to as "compressed" signals.

Various techniques are used to perform the compression. These compression techniques are not compatible with one another, in that a file produced using one compression technique cannot be expanded or decompressed using another technique. Some of the more common techniques include Adaptive Pulse Code Modulation (APCM), μ-law, and Run-Length Encoding (RLE). Generally, a file does not contain a mixture of compressed and uncompressed data, nor data produced by a mixture of compression techniques.

Some digital signal devices incorporate a hardware component known as a digital signal processor (DSP) and its associated memory and control circuitry. Hereinafter, a DSP and its associated components are referred to collectively as a "DSP". These devices can use the DSP to compress and decompress (COompress/DECompress or "CODEC") digital signals; this is commonly referred to as a "hardware CODEC" technique. DSP-equipped devices can also process uncompressed digital signals, but this wastes resources, since DSP-equipped devices are more expensive than devices without DSPs.

Devices without a DSP cannot process most compressed digital signals, and thus generally only operate with uncompressed digital signals. If such a device must operate with a compressed file, the CODEC function must be performed by software running on a central processing unit (CPU) in the associated computer before the device stores or retrieves the data from the file; this intermediate CODEC function is commonly referred to as a "software CODEC" technique.

Mismatches sometimes occur between a file's data type, uncompressed or compressed, and a device's ability to perform a hardware CODEC function. When these mismatches occur, they lead to one of two major problems: degraded system performance or poor resource utilization. Specifically, a compressed file being processed by a

BNSDOCID: <EP 0685824A2 I >

device that lacks hardware CODEC capabilities requires the CPU to perform extra processing, i.e. software compression/decompression, and so negatively impacts overall system performance. On the other hand, an uncompressed file being processed by a device that possesses hardware CODEC capabilities underutilizes the relatively expensive DSP and wastes storage space and I/O bandwidth, since a compressed file would take much less space.

Until now, creators of commercial digital signal files generally were compelled to supply uncompressed files in order to ensure maximum compatibility with the range of available devices. This consumed large amounts of file storage space; it also underutilized the DSP on systems with devices capable of performing a hardware CODEC function.

In a manner similar to digital signal devices, applications that manipulate digital signals must be able store and retrieve digital signals from files where the data may have been compressed using one of many compression techniques. In prior art systems, in order to store or retrieve data from a compressed file, an application must contain a software CODEC routine that matches the file's compression type.

Further, data is commonly structured when it is stored in a file. Specifically, the structure accords with a "file format", i.e. a specification of the layout of data words in the file's data portion and the content and layout of a file header. File formats are independent of the file contents, moreover applications may create several files, all containing the same type of data, but using different file formats. Files are often formatted differently under different operating systems, for example MS-DOS and OS/2 ®. Thus, each software CODEC routine must be written to compress and decompress data using both a particular compression technique and a particular file format. Since there are numerous compression types and file formats, each application must contain many such software CODEC routines in order to be compatible with as many compression types and file formats as possible.

However, applications that contain many CODEC routines have program size and complexity problems. Large or complex programs are more difficult to write, debug, and maintain. Each application writer must duplicate CODEC routines that already exist in other applications, introducing the problem of duplicated effort. New compression techniques are constantly being developed, so updating existing applications is also problematic. On the other hand, an application with only one or a small number of CODEC routines has a disadvantage in that it can process only a limited number of compression types and file formats.

Some prior art systems, therefore, support multiple compression types. These systems fall into two groups: "hardware-only" CODEC systems and "software-only" CODEC systems. Hardware-only CODEC systems ensure that devices will be able to process any file by always requiring a DSP. These systems have two disadvantages. They preclude the use of less expensive devices, ones without a DSP, and since they do not compress or decompress data transferred between applications and files, they fail to solve the application size and complexity problem.

There are also software-only CODEC systems, which perform all CODEC functions in software, regardless of whether a DSP is available. The file I/O handler in these systems typically detects a file's format and which, if any, compression technique was used by reading the file header upon opening the file. The file I/O handler maintains a library of CODEC routines. Using the appropriate routine, the file I/O handler passes uncompressed data to and from the application or device, regardless of the file's compression type. However, when a DSP-equipped device is available, these systems waste valuable CPU time performing software CODEC functions, and therefore negatively impact overall system performance.

Consequently, neither hardware-only nor software-only CODEC systems solves the problems of resource utilization and application size and complexity.

SUMMARY OF THE INVENTION

Accordingly, the present invention provides a method for transferring data including compressed data to utilization means in response to a request, the method comprising the steps of: determining whether the utilization means contains hardware for decompressing the compressed data; ascertaining a compression technique used to compress the compressed data; transferring the compressed data to the utilization means; using a software routine to decompress the compressed data when the utilization means does not contain the hardware; and controlling the hardware to decompress the compressed data in accordance with the compression technique when the utilization means does contain the hardware.

According to another aspect, the invention also provides a data transfer system for transferring data including compressed data to utilization means in response to a request, the system comprising: means responsive to the request for determining whether the utilization means contains hardware for decompressing the compressed data; means responsive to the request for ascertaining a compression technique used to compress the compressed data; a file I/O handler responsive to the request for transferring the data to the utilization

means, the file I/O handler being controllable to decompress the compressed data while transferring the data to the utilization means; means cooperating with the determining means and the ascertaining means for controlling the file I/O handler to decompress the data when the utilization means does not contain the hardware; and means for controlling the hardware to decompress the compressed data in accordance with the compression technique when the utilization means does contain the hardware.

Systems according to the invention can translate between a common, uncompressed data format and any of several compressed data formats. These systems dynamically choose to perform the translation in software or in hardware. These systems base the choice on the particular data compression format used, and on the presence or absence of hardware translation resources.

This is achieved by dynamically choosing between hardware and software CODEC techniques based on the hardware resources available and on each file's compression type. The invention further provides a common, uncompressed data interchange format between applications and data files, regardless of the compression technique or file format used to create the data file.

More specifically, a preferred embodiment of the invention employs a digital signal manager positioned between the file I/O handler and digital signal devices or application programs. In response to a file's compression type, as indicated in the file's header, and the characteristics of the digital signal device, as stored in a "resource file" or as sensed by the digital signal manager or by interrogating the device directly, the digital signal manager dynamically selects either a hardware or software CODEC technique for transfers between the digital signal device and the file. The digital signal manager also provides a common data interchange format to the application programs, which in accordance with an illustrative embodiment, is uncompressed data. Thus, each application need only be concerned with a single compression type and file format.

More particularly, in response to an application's request to open a file, if an existing file contains compressed data, or a new file is to contain compressed data, the digital signal manager instructs the file I/O handler to invoke the appropriate decompression routine when the application retrieves data from the file, and to invoke the appropriate compression routine when the application stores data in the file. If the file contains uncompressed data, or a new file is to contain compressed data, the digital signal manager instructs the file I/O handler to pass the data unmodified between the application and the file. That is, the digital signal manager makes all files appear to the application as though they were uncompressed. This common, uncompressed, data format provides compatibility between substantially all applications and substantially all digital signal files, regardless of which, if any, compression techniques were used to create the files.

The digital signal manager maintains resource files that contain selected operating characteristics of each available digital signal device. The resource files may be supplied, for example, by the device vendors. These characteristics include information concerning the device's operating mode(s), including, for each operating mode, the sampling rate, i.e. the number of samples per second taken by the device; the precision, i.e. the number of data bits per sample; and the number of analog channels used. The resource files also include additional information, such as the presence and quantity in the device of various hardware components useful for data compression or decompression, for example a DSP and the amount of memory available to the DSP.

In response to an application's request to establish a file handle between an existing file and a digital signal device, the digital signal manager uses the file I/O handler to read the file's header and to ascertain the file's characteristics. These characteristics include the sampling rate, precision, number of channels, and which compression technique, if any, was used to create the file. If the application specifies a new file, then the application may specify the file's characteristics, although the file I/O handler specifies default characteristics. The digital signal manager uses both the file's characteristics and the device's hardware characteristics, as reflected in the resource file, to decide whether the device can process the data directly, or whether the data must be compressed or decompressed by a software CODEC routine first.

The digital signal manager then controls the file I/O handler to transfer the data in an appropriate manner. For example, if the digital signal manager ascertains that the device can directly process the data in the file, the digital signal manager adjusts the device's operating characteristics to match as closely as possible the characteristics under which the data file was created. The digital signal manager then instructs the file I/O handler to pass the data directly between the device and the file, that is, without the file I/O handler compressing or decompressing the data.

Alternatively, if the digital signal manager ascertains that the data must be compressed or decompressed before the transfer, the digital signal manager instructs the file I/O handler to invoke the appropriate decompression routine when the device reads from the file, and the appropriate com-

BNSDOCID: <EP 0685824A2 I >

pression routine when the device writes to the file; the file I/O handler performs software compression/decompression.

The use of the digital signal manager to decide whether the data must be compressed or decompressed by a software CODEC routine enhances resource utilization. A DSP is utilized whenever possible to process the data, freeing the central processing unit (CPU) to perform other functions. The CPU performs software compression/decompression only when the device is incapable of processing the data directly, yet devices without DSPs are not limited to operating with uncompressed data files.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and further advantages of the invention may be better understood by referring to the following description in conjunction with the accompanying drawings, in which:

Fig. 1 is a schematic block diagram of a computer on which an embodiment of the invention can be practised;

Fig. 2 is a schematic block diagram showing a data transfer system according to the invention in relation to other software and hardware components with which it interacts;

Fig. 3 is a flow chart, which illustrates aspects of a method according to the invention related to the system illustrated in Fig. 2;

Fig. 4 is a flow chart, which illustrates a method according to the invention in relation to the system illustrated in Fig. 2; and

Figs. 5A , 5B and 5C combine to form a flow chart, which illustrates in more detail a portion of the method illustrated in Fig. 4.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

The invention is preferably practised in the context of an operating system resident on a personal computer, such as the system represented in Fig. 1. The computer 100 is controlled by a central processing unit (CPU) 102, which may be a conventional microprocessor, and a number of other units, which are provided to accomplish specific tasks. A system bus 104 interconnects the CPU 102 and the other units. Although a particular computer may only have some of the units illustrated in Fig. 1, or may have additional components not shown, most computers will include at least the units shown. Specifically, the computer 100 includes a read-only memory (ROM) 106 for permanent storage of the computer's configuration and basic operating commands; a random access memory (RAM) 108 for temporary storage of instructions and data; an input/output (I/O) adapter 110 for connecting peripheral devices such as a disk unit 112; a keyboard adapter 114 for connecting a keyboard 116; a display adapter 118 for connecting a display 120; and a digital signal device, which in this example is a sound board 122 for connecting a speaker 124 and a microphone 126. Hereinafter, the ROM 106, RAM 108, and disk unit 112 collectively are referred to as the "data storage units".

Fig. 2 depicts various files containing programs and data and resident in the various data storage units in the system. The system may contain one or more digital signal devices, for example 200a, 200b, and 200c. Optionally, one or more devices, such as 200c, may incorporate a digital signal processor (DSP) 202. In the preferred embodiment of the invention, the vendor who supplies a device also supplies a resource file, for example 204a, 204b, or 204c, corresponding to the device 200a, 200b, or 200c, respectively. Hereinafter, the resource file is referred to as "resource file 204". Each resource file 204 lists, for example, whether the associated device 200 incorporates a DSP, the amount of memory associated with the DSP, the characteristics of the device 200 that can be adjusted and the range of values of these characteristics. The use of the resource file 204 is described in more detail below.

The system contains one or more applications, for example 206a, 206b, and 206c, each resident in the RAM 108 or ROM 106 of Fig. 1. Hereinafter, the application is referred to as "application 206".

Digital signal files 208, 210a, 210b, and 210c contain digital signals. File 208 contains uncompressed data. Files 210a, 210b, and 210c, on the other hand, contain data compressed according to different compression techniques, although more than three such techniques exist. Each file contains a data portion containing digital signals and a header portion specifying characteristics of the data portion, e.g. sampling rate, bits per sample and compression format, i.e. uncompressed or compression technique. The application 206 or the device 200 interacts with a file 208 or 210 by means of a file input/output (I/O) handler 212. The file I/O handler 212 is part of the computer operating system 214, and it manages the administrative operations of opening and closing the files and actually storing and retrieving data in a storage device. The operating system 214 and its file I/O handler 212 reside in the RAM 108 and/or ROM 106 of Fig. 1.

An application 206 can perform two types of operations. In the first type, the application 206 reads and/or writes a digital signal in one or more files 208 and/or 210 in order to perform a non-real-time operation, for example to cut a segment of audio data from one file and paste it into another

file. In the second type of operation, the application 206 instructs the device 200 to perform a real-time operation, for example record or play back, and the device 200 then reads or writes a digital signal in a file 208 or 210. These two types of operation are described in detail below.

When performing an operation of the first type, non-real-time, the application 206 sends a request to the operating system 214, i.e. to a digital signal manager 216. The request identifies the file 208 or 210 to be processed. The digital signal manager 216 performs the operations generally illustrated in the flow chart in Fig. 3. Thus the digital signal manager 216 begins at step 300 and sends a request at step 302 to the file I/O handler 212 to open the file 208 or 210 and to detect which, if any, compression technique was used to create the file, and in which format the file exists.

The file I/O handler 212 may contain an "I/O procedure", for example 218a, 218b, or 218c, for each combination of file format and storage medium that the system supports. Each I/O procedure is capable of storing data in and retrieving data from a file, and translating between the file's format and a common format. The file I/O handler 212 interrogates the I/O procedures until it finds one that recognizes the file 208 or 210; the file I/O handler uses this I/O procedure for all transfers to and from the file 208 or 210. This arrangement, which is not part of the present invention, is described in more detail in commonly assigned United States patent application serial number 07/960,976, filed on December 31, 1991 by D. M. Dorrance, et al, for a Data Processing File Format Transparency System, which is hereby incorporated by reference. The file I/O handler 212 also detects which, if any, compression technique was used to create the file 208 or 210 by reading the file's header, and provides this information to the digital signal manager 216.

At step 304 the digital signal manager 216 examines the response from the file I/O handler 212. If the file is uncompressed, the digital signal manager 216 establishes at step 306 a file handle 220 between the file I/O handler 212 and the application 206. A file handle is sometimes referred to as a "channel", and it is well known how to establish a file handle. See, for example, Multimedia Presentation Manager Programmer Reference, IBM order number 71G2222. The application 206 then reads and/or writes in the file 208. The digital signal manager 216 finishes at step 308.

If the file is compressed, the file I/O handler 212 loads into the file I/O handler the appropriate compression type-specific compression/decompression (COmpression/DECompression or CODEC) routine, for example 222a, 222b, or 222c. This process, which is not part of the present invention, is described in more detail in commonly assigned United States patent application serial number 07/981040, filed on November 24, 1992 by Fetchi Chen and Daniel Michael Dorrance, for a Software Mechanism for Providing CODEC Transparency, which is hereby incorporated by reference. The digital signal manager 216 establishes at step 310 a file handle 220 between the file I/O handler 212 and the application 206. The digital signal manager 216 instructs the file I/O handler 212 to use the appropriate CODEC routine 222 for the data transfers between the application 206 and the file 210. The CODEC routine 222 decompresses the data as the application 206 reads from the file 210 and compresses the data as the application 206 writes to the file 210. The digital signal manager 216 finishes at step 308.

Thus, the digital signal manager 216 ensures that the application 206 exchanges data with the file 208 or 210 in a common, uncompressed format.

In the second type of operation, the application 206 instructs a device 200 to perform a real-time operation. The flow chart in Fig. 4 illustrates generally the operation of the digital signal manager 216 when it performs an operation of the this type. The application 206 sends a request to the operating system 214, i.e. to the digital signal manager 216. The request indicates that the application 206 directs a device 200 to read or write a digital signal in a file 208 or 210.

The digital signal manager 216 starts at step 400. At step 402, it ascertains which characteristics of the device 200 can be adjusted and the range of values these characteristics can take on. In the preferred embodiment, the digital signal manager 216 ascertains the device's characteristics, step 402, by reading the resource file 204. In an alternate embodiment, the digital signal manager 216 ascertains the device's characteristics by interrogating the device 118. These characteristics include the number of samples per second; the precision, i.e. number of data bits per sample; and the number of channels used. Also at step 402, the digital signal manager 216 determines whether the device 200 has a DSP 202, and if so, the amount of memory available to the DSP.

At step 404, the digital signal manager 216 sends a request to the file I/O handler 212 to open the file 208 or 210 and to detect which, if any, compression technique was used to create the file. At step 406 the digital signal manager 216 examines the response from the file I/O handler 212. If the file is uncompressed, at step 408 the digital signal manager 216 sets the device's adjustable characteristics, thereby instructing the device 200 to process an uncompressed digital signal. The digital signal manager 216 sets the device's adjust-

BNSDOCID: <EP_____0685824A2_I_>

able characteristics by well known means, for example by setting and clearing the appropriate bits in the device's control registers. The digital signal manager 216 establishes at step 410 a file handle 224 between the file I/O handler 212 and the device 200. The device 200 then reads from and/or writes to the file 208. The digital signal manager 216 finishes the operation at step 412.

If the file is compressed, at step 414 the digital signal manager 216 determines whether the device 200 has a DSP 202 and sufficient memory to process the digital signal. If the device 200 does not have a DSP or does not have sufficient memory, the digital signal manager 216 establishes at step 416 a file handle 224 between the file I/O handler 212 and the device 200. The digital signal manager 216 instructs the file I/O handler 212 to use a CODEC routine 222 for the data transfers between the device 200 and the file 210. The CODEC routine 222 decompresses the data as the device 200 reads from the file 210 and compresses the data as the device 200 writes to the file 210. Although the file 210 is actually compressed, the device 200 will then process uncompressed data. Therefore, at step 418 the digital signal manager 216 sets the device's adjustable characteristics, thereby instructing the device 200 to process an uncompressed digital signal. The digital signal manager 216 finishes at step 412.

If the device 200 has a DSP and sufficient memory to process the digital signal, the digital signal manager 216 at step 420 calculates the appropriate values for the device's adjustable characteristics. The digital signal manager 216 matches these values as closely as possible to the characteristics under which the digital signal file was created. The digital signal manager 216 calculates the number of bits per sample, the number of channels, and the sampling rate (the number of samples per second) to be used by the device 200.

The flow charts of Fig. 5A, 5B, and 5C illustrate generally the operations of step 420. Referring first to Fig. 5A, the digital signal manager 216 starts the number-of-bits-per-sample calculation at step 500, and at step 502 it sets the "current approximation" to the number of bits per sample in the device's first mode of operation. At step 504 it tests whether the device 200 has any more modes of operation to consider. If there are no more modes to consider, at step 506 the digital signal manager sets the "number of bits per sample closest match" to the "current approximation" and at step 508 it transfers to the flowchart in Fig. 5B.

If there are more modes of operation to consider, the digital signal manager 216 at step 510 sets "X" to the number of bits per sample in the device's next mode of operation. At step 512 the digital signal manager 216 sets "A" to the absolute value of the difference between the file's 210 number of bits per sample and the "current approximation". At step 514 the digital signal manager 216 sets "B" to the absolute value of the difference between the file's 210 number of bits per sample and "X". At step 516 the digital signal manager 216 compares "A" with "B". If "B" is greater than "A" the digital signal manager 216 loops directly back to step 504. Otherwise, at step 518 it sets the "current approximation" to "X" and loops back to step 504.

In Fig. 5B, the digital signal manager 216 starts the number-of-channels calculation at step 600, and at step 602 it sets the "current approximation" to the number of channels in the device's first mode of operation. At step 604 it tests whether the device 200 has any more modes of operation to consider. If there are no more modes to consider, at step 606 the digital signal manager sets the "number of channels closest match" to the "current approximation" and at step 608 it transfers to the flowchart in Fig. 5C.

If there are more modes of operation to consider, the digital signal manager 216 at step 610 sets "X" to the number of channels in the device's next mode of operation. At step 612 the digital signal manager 216 sets "A" to the absolute value of the difference between the file's 210 number of channels and the "current approximation". At step 614 the digital signal manager 216 sets "B" to the absolute value of the difference between the file's 210 number of channels and "X". At step 616 the digital signal manager 216 compares "A" with "B". If "B" is greater than "A" the digital signal manager 216 loops directly back to step 604. Otherwise, at step 618 it sets the "current approximation" to "X" and loops back to step 604.

In Fig. 5C, the digital signal manager 216 starts the sampling rate calculation at step 700, and at step 702 it sets the "current approximation" to the sampling rate in the device's first mode of operation. At step 704 it tests whether the device 200 has any more modes of operation to consider. If there are no more modes to consider, at step 706 the digital signal manager sets the "sampling rate closest match" to the "current approximation" and at step 708 it finishes.

If there are more modes of operation to consider, the digital signal manager 216 at step 710 sets "X" to the sampling rate in the device's next mode of operation. At step 712 the digital signal manager 216 sets "A" to the absolute value of the difference between the file's 210 sampling rate and the "current approximation". At step 714 the digital signal manager 216 sets "B" to the absolute value of the difference between the file's 210 sampling rate and "X". At step 716 the digital signal manager 216 compares "A" with "B". If "B" is greater

than "A" the digital signal manager 216 loops back to step 704. Otherwise, at step 718 it sets the "current approximation" to "X" and loops directly back to step 704.

At step 422 in Fig. 4 the digital signal manager 216 sets the device's adjustable characteristics so they are consistent with those calculated at step 420. The digital signal manager 216 establishes at step 410 a file handle 224 between the file I/O handler 212 and the device 200. The device 200 then reads and writes to the file 208. The device 200 performs the CODEC function; this is called "hardware CODEC". The digital signal manager 216 finishes at step 412.

Thus, the digital signal manager 216 dynamically chooses between hardware and software CODEC techniques and in so doing it maximizes the usage of the digital signal devices' hardware capabilities. That is, it uses hardware CODEC techniques whenever they can be used in processing the digitized signals and it uses software CODEC routines whenever hardware CODEC techniques cannot be used. It thus provides for the operation of non-compression-specific application programs while optimizing the overall efficiency of the system in dealing with digital signal files.

Claims

1. A method for transferring data including compressed data (210) to utilization means (200) in response to a request, the method comprising the steps of:

   determining (402) whether the utilization means (200) contains hardware (202) for decompressing the compressed data;

   ascertaining (404, 406) a compression technique used to compress the compressed data;

   transferring (410, 416) the compressed data to the utilization means;

   using a software routine (222) to decompress the compressed data when the utilization means does not contain the hardware; and

   controlling (422) the hardware (202) to decompress the compressed data in accordance with the compression technique when the utilization means does contain the hardware.

2. A method as claimed in claim 1 in which said determining step comprises the further steps of:

   obtaining a resource file (204) containing characteristics of the utilization means; and

   reading the resource file to determine whether the utilization means contains the hardware.

3. A method as claimed in either claim 1 or claim 2 in which said ascertaining step comprises reading information relating to the compression technique from header information in a file of compressed data to be transferred.

4. A method as claimed in any preceding claim including the steps of determining (304, 406) whether data to be transferred is compressed or uncompressed and transferring uncompressed data directly to said utilization means.

5. A method as claimed in claim 4 for use in a system in which said utilization means includes a digital signal device (200) and which system includes an application program (206) for issuing requests for data transfer either to said digital signal device or to the application program itself, the method including the further step, where data to be transferred to the application program is compressed, of using (310) a software routine (222) to decompress that data.

6. A method as claimed in any preceding claim which includes the further step, in response to a determination that the utilization means contains said decompression hardware and that the data to be transferred is compressed, of setting the characteristics of the utilization means and decompression hardware to match characteristics of the compressed data.

7. A method as claimed in claim 1 in which said determining step comprises directly interrogating the utilization means in order to determine whether the utilization means contains the hardware.

8. A data transfer system for transferring data including compressed data (210) to utilization means (200) in response to a request, the system comprising:

   means (402) responsive to the request for determining whether the utilization means contains hardware (202) for decompressing the compressed data;

   means (404, 406) responsive to the request for ascertaining a compression technique used to compress the compressed data;

   a file I/O handler (212) responsive to the request for transferring the data to the utilization means, the file I/O handler being controllable to decompress the compressed data while transferring the data to the utilization means;

   means (416) cooperating with the determining means and the ascertaining means for

BNSDOCID: <EP_____0685824A2_I_>

controlling the file I/O handler to decompress the data when the utilization means does not contain the hardware; and

means (422) for controlling the hardware to decompress the compressed data in accordance with the compression technique when the utilization means does contain the hardware.

9. A system as claimed in claim 8 further comprising a resource file (204) containing characteristics of the utilization means and wherein the determining means is arranged to read the resource file to determine whether the utilization means contains the hardware.

10. A system as claimed in either claim 8 or claim 9 in which the ascertaining means is arranged to read information relating to the compression technique from header information in a file of compressed data to be transferred.

11. A system as claimed in any one of claims 8 to 10 wherein the utilization means comprises a digital signal device, the digital signal device including said decompression hardware.

12. A system as claimed in any one of claims 8 to 11 including means responsive to said determining means (402) determining that the utilization means contains said decompression hardware and to a determination that the data to be transferred is compressed for setting the characteristics of the utilization means and decompression hardware to match characteristics of the compressed data.

13. A system as claimed in any one of claims 8 to 12 including means (304, 406) for determining whether data to be transferred is compressed or uncompressed and an application program (206) for issuing requests for data transfer either to said utilization means or to the application program itself, in which the system further includes means (310) responsive to requests for data transfer to the application program itself and to a determination (304) that the data is compressed to control the file I/O handler (212) to decompress (222) that data.

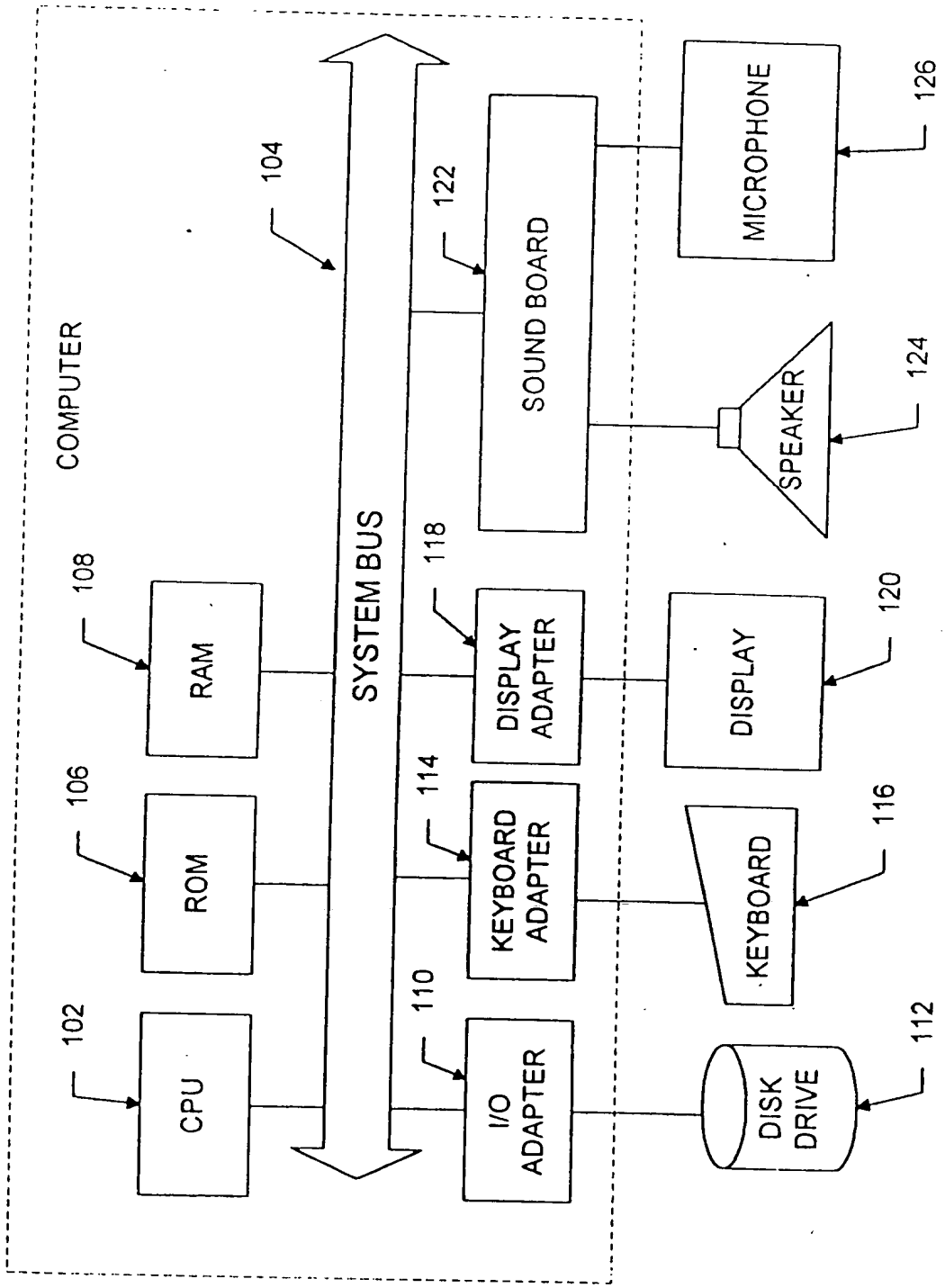

FIG. 1

BNSDOCID: <EP_____0685824A2_I_>

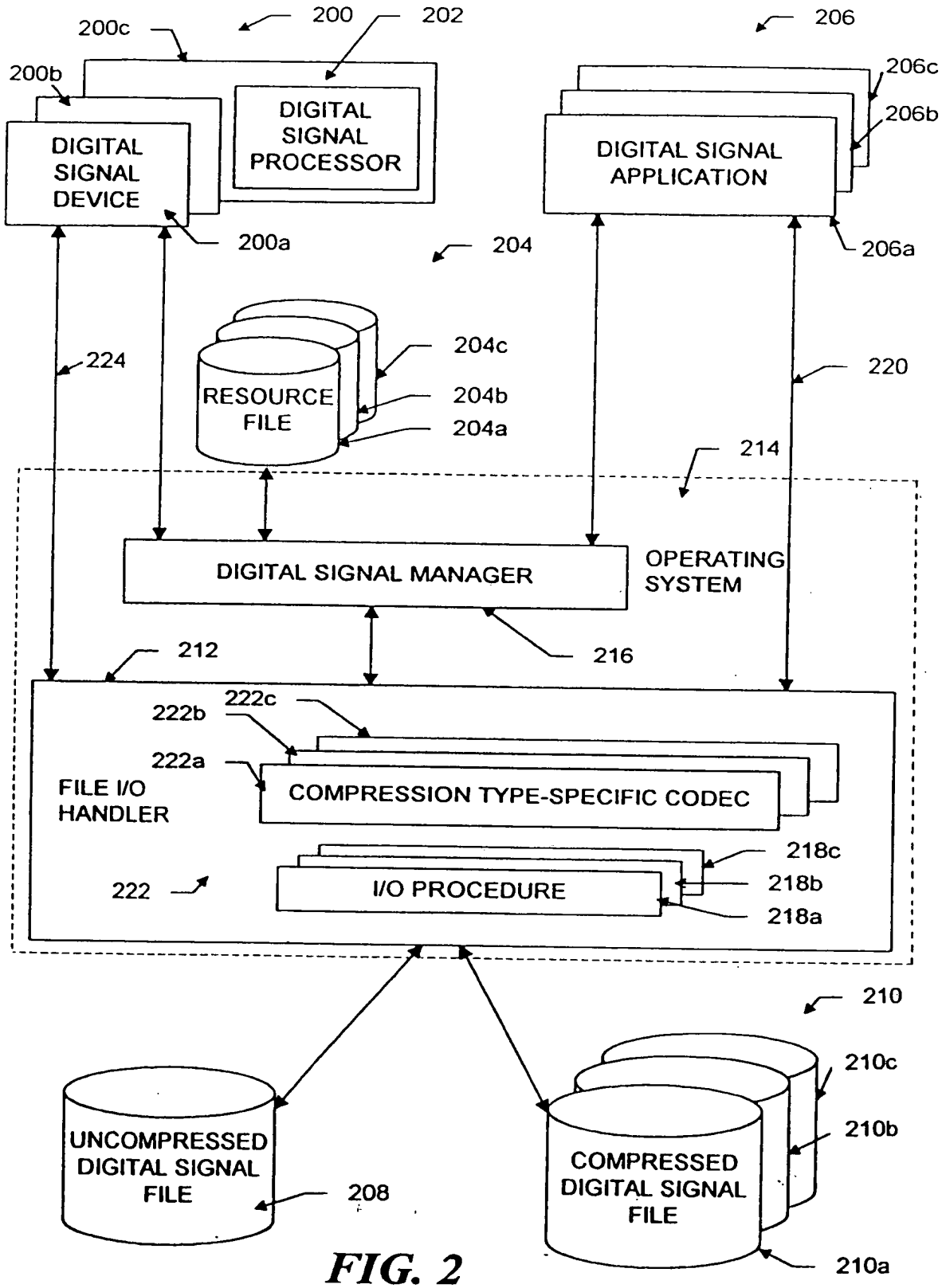
200
200c
202
200b
DIGITAL SIGNAL DEVICE
DIGITAL SIGNAL PROCESSOR
200a
206
206c
206b
DIGITAL SIGNAL APPLICATION
206a
204
224
RESOURCE FILE
204c
204b
204a
220
214
DIGITAL SIGNAL MANAGER
OPERATING SYSTEM
212
216
222b
222c
222a
FILE I/O HANDLER
COMPRESSION TYPE-SPECIFIC CODEC
218c
222
I/O PROCEDURE
218b
218a
210
UNCOMPRESSED DIGITAL SIGNAL FILE
208
210c
210b
COMPRESSED DIGITAL SIGNAL FILE
210a

FIG. 2

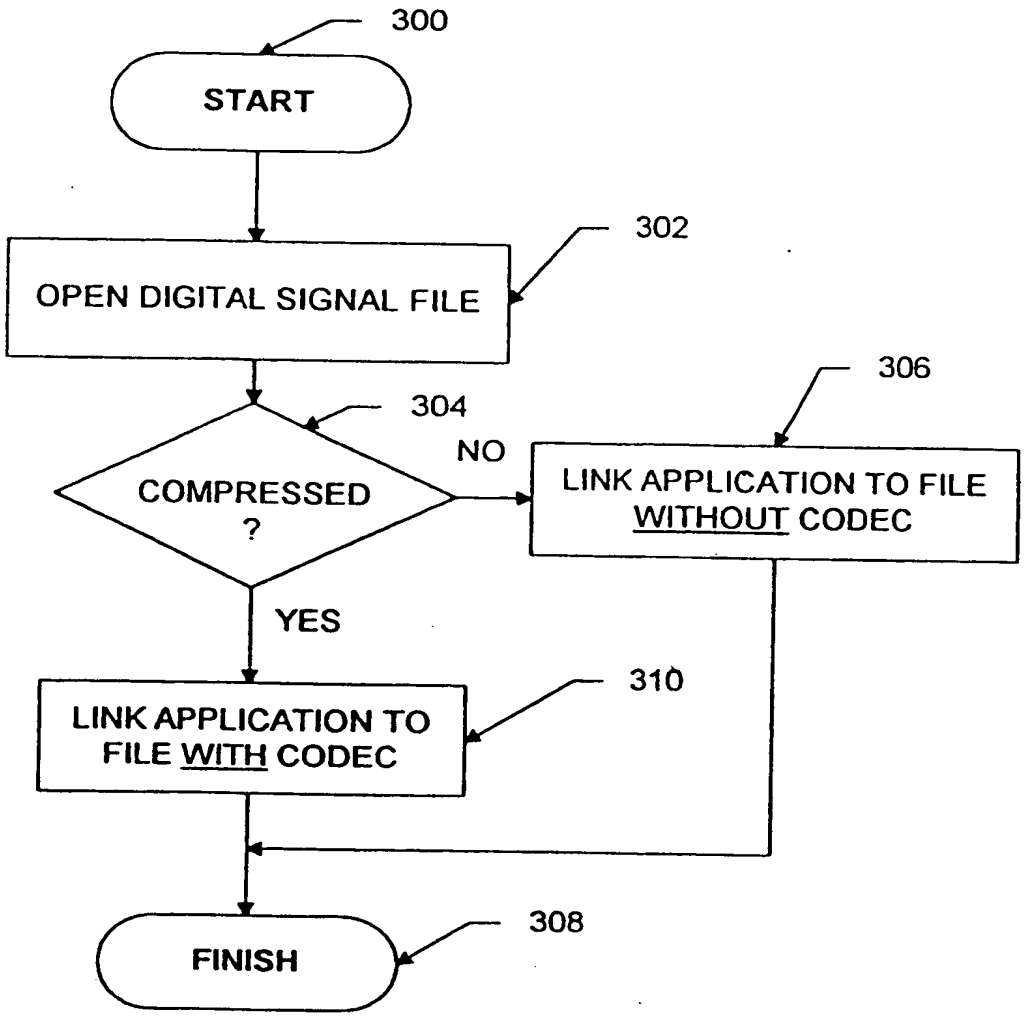

FIG. 3

BNSDOCID: <EP_____0685824A2_I_>

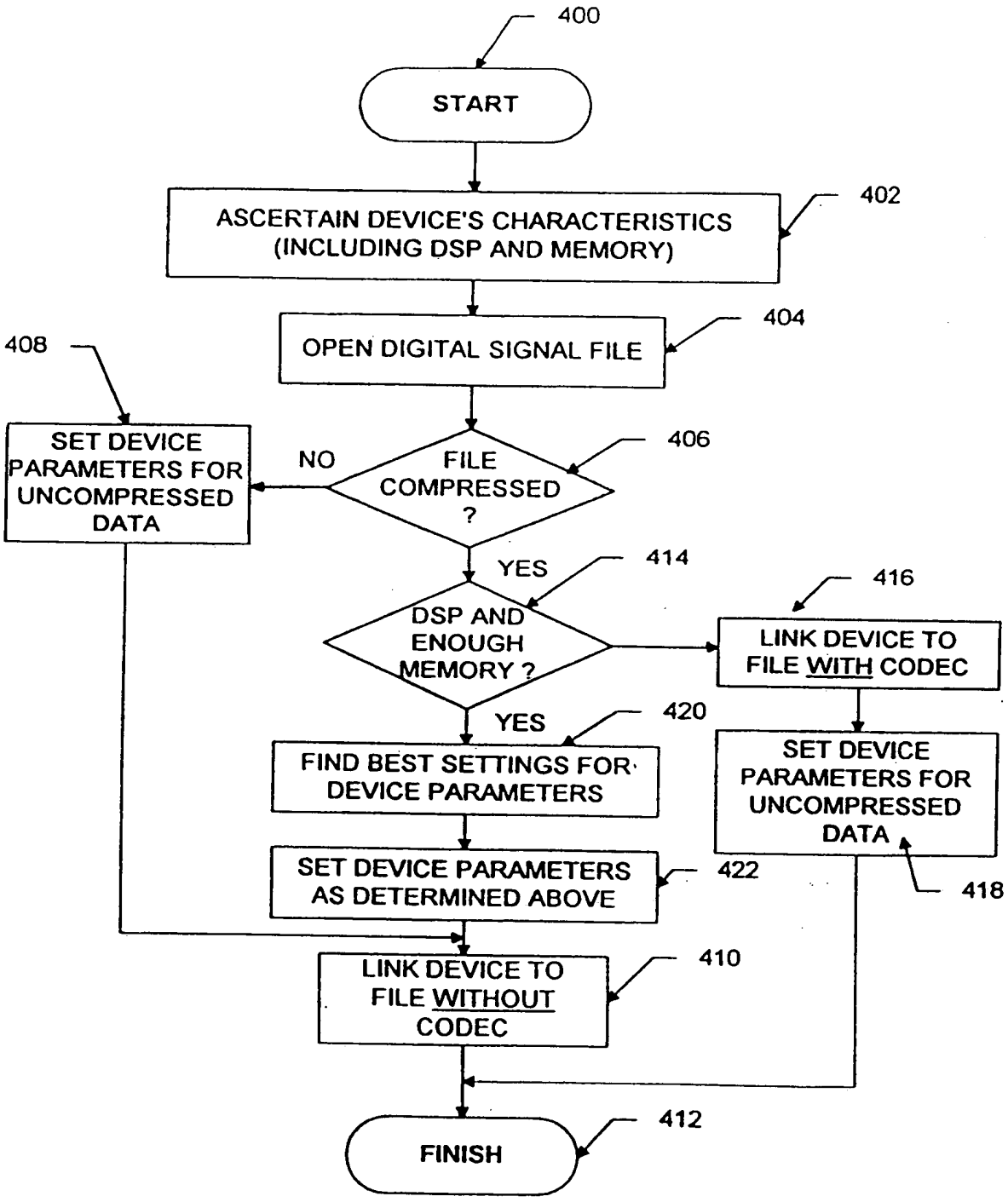
400
START
ASCERTAIN DEVICE'S CHARACTERISTICS (INCLUDING DSP AND MEMORY)
402
OPEN DIGITAL SIGNAL FILE
404
408
SET DEVICE PARAMETERS FOR UNCOMPRESSED DATA
NO
FILE COMPRESSED ?
406
YES
414
DSP AND ENOUGH MEMORY ?
416
LINK DEVICE TO FILE WITH CODEC
YES
420
FIND BEST SETTINGS FOR DEVICE PARAMETERS
SET DEVICE PARAMETERS FOR UNCOMPRESSED DATA
SET DEVICE PARAMETERS AS DETERMINED ABOVE
422
418
LINK DEVICE TO FILE WITHOUT CODEC
410
FINISH
412

FIG. 4

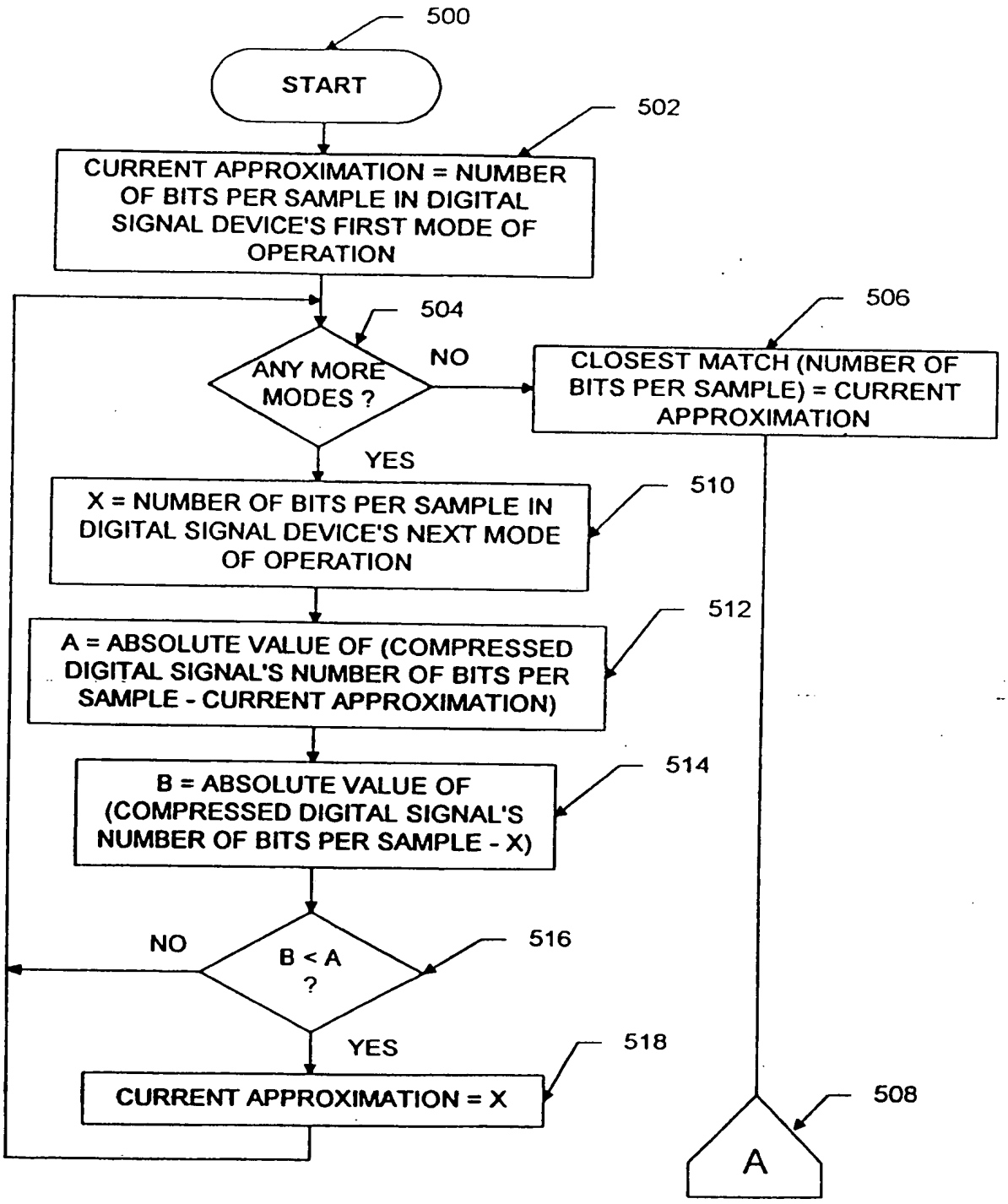

FIG. 5A

BNSDOCID: <EP 0685824A2 >

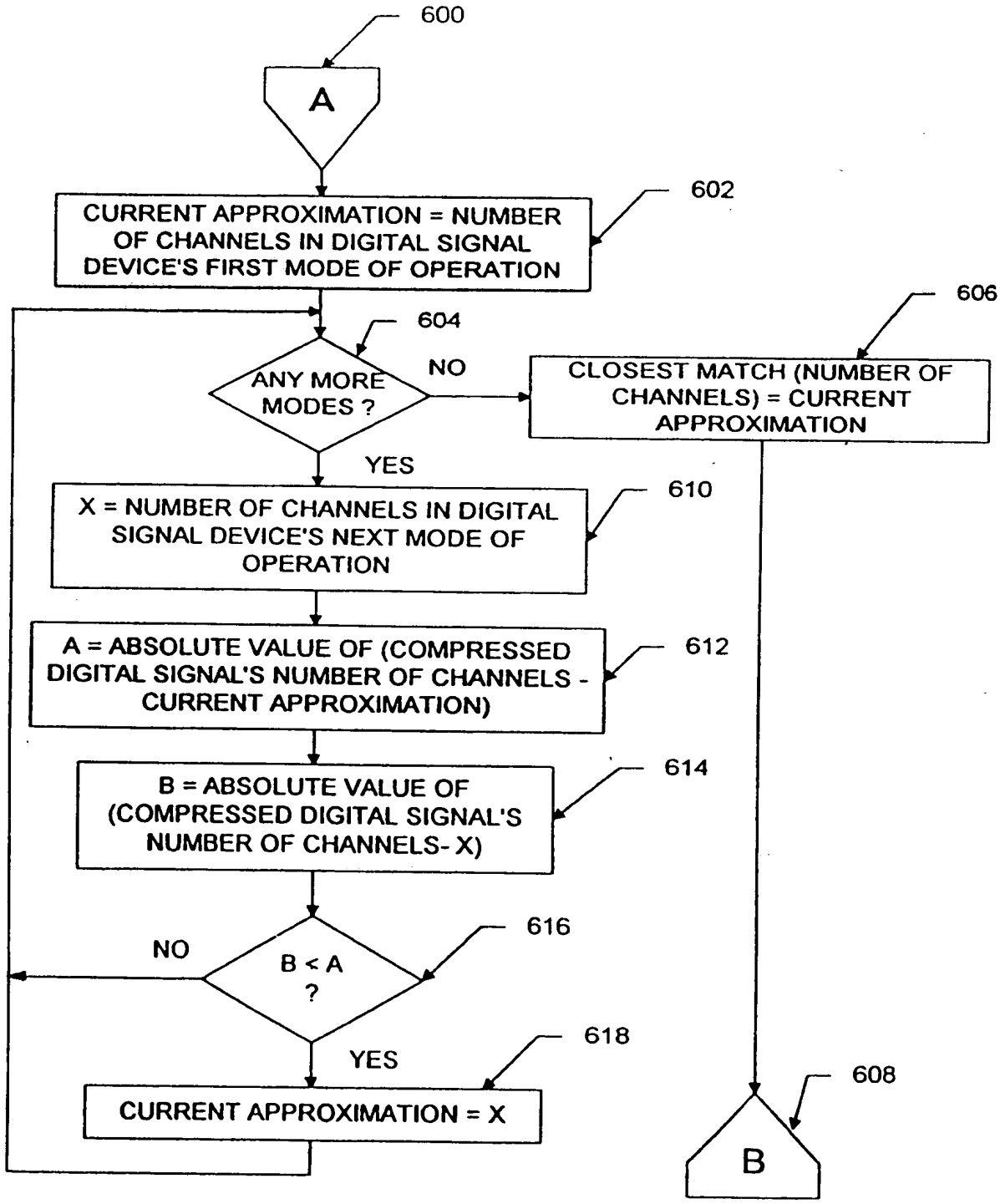
600
A
CURRENT APPROXIMATION = NUMBER OF CHANNELS IN DIGITAL SIGNAL DEVICE'S FIRST MODE OF OPERATION
602
606
604
ANY MORE MODES ?
NO
CLOSEST MATCH (NUMBER OF CHANNELS) = CURRENT APPROXIMATION
YES
X = NUMBER OF CHANNELS IN DIGITAL SIGNAL DEVICE'S NEXT MODE OF OPERATION
610
A = ABSOLUTE VALUE OF (COMPRESSED DIGITAL SIGNAL'S NUMBER OF CHANNELS - CURRENT APPROXIMATION)
612
B = ABSOLUTE VALUE OF (COMPRESSED DIGITAL SIGNAL'S NUMBER OF CHANNELS- X)
614
NO
B < A ?
616
YES
618
CURRENT APPROXIMATION = X
608
B

FIG. 5B

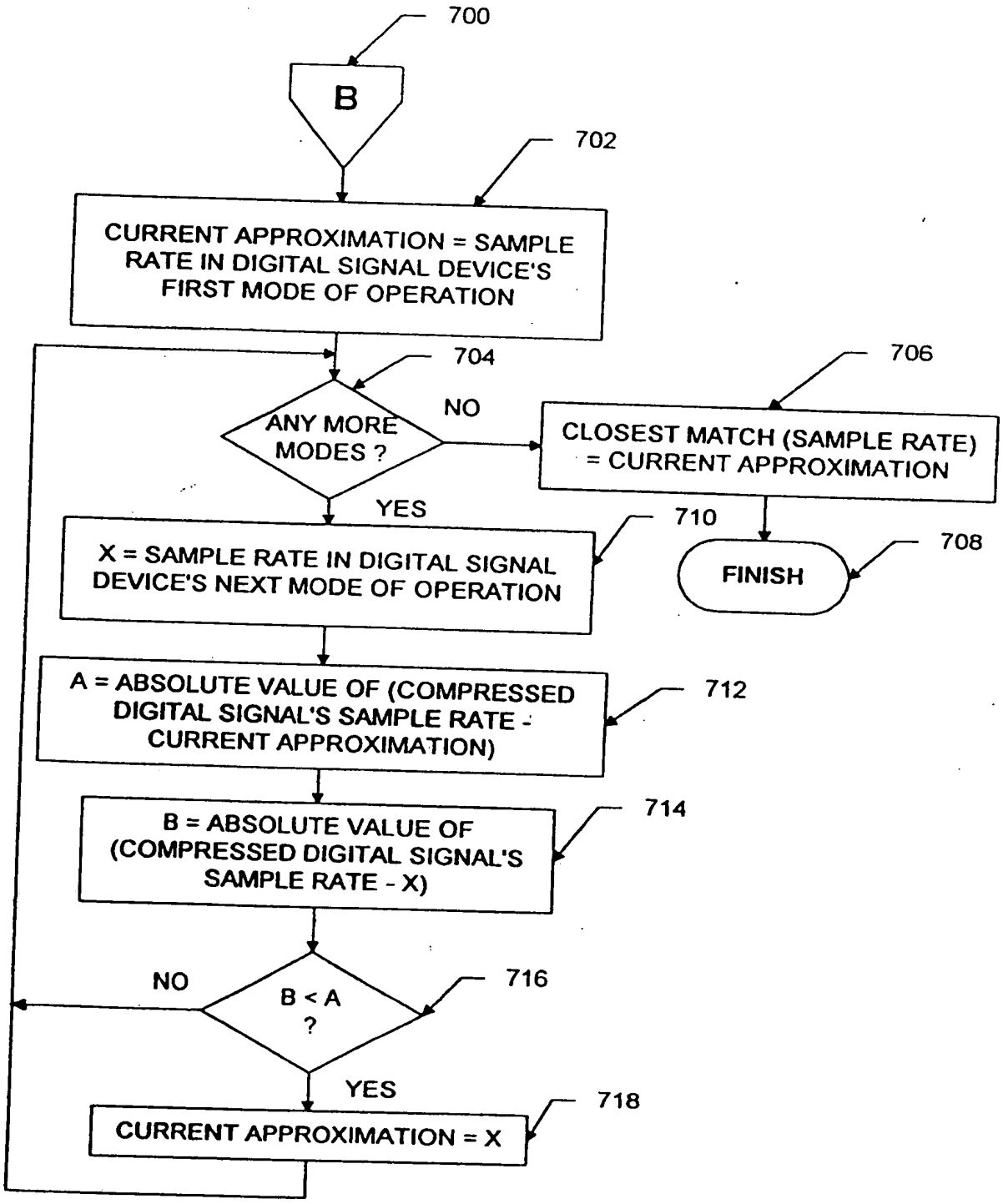

FIG. 5C

RNSDOCID: -EP 0685824A2 I >